

# soc2seq: Social Embedding meets Conversation Model

Parminder Bhatia  
Yik Yak  
3560 Piedmont Rd NE  
Atlanta, GA, 30305, USA  
parminder.bhatia243@gmail.com

Marsal Gavalda  
Yik Yak  
3560 Piedmont Rd NE  
Atlanta, GA, 30305, USA  
marsal@yikyakapp.com

Arash Einolghozati  
Yik Yak  
3560 Piedmont Rd NE  
Atlanta, GA, 30305, USA  
einolghozati@gmail.com

## ABSTRACT

While liking or upvoting a post on a mobile app is easy to do, replying with a written note is much more difficult, due to both the cognitive load of coming up with a meaningful response as well as the mechanics of entering the text. Here we present a novel textual reply generation model that goes beyond the current auto-reply and predictive text entry models by taking into account the content preferences of the user, the idiosyncrasies of their conversational style, and even the structure of their social graph. Specifically, we have developed two types of models for personalized user interactions: a **content-based conversation model**, which makes use of location together with user information, and a **social-graph-based conversation model**, which combines content-based conversation models with social graphs.

## 1. INTRODUCTION

Yik Yak is a location-based social app, where people can view text and images posted within a five-mile radius. Users talk about a variety of topics on our platform such as sports, politics, entertainment, food, etc. They voice their opinions on these topics in the form of yaks (posts), comments (replies to yaks), and upvotes and downvotes to both yaks and comments.

This user activity results in a flood of events that we then analyze for a variety of purposes, such as categorizing content, building user profiles, and deriving a social graph, to ultimately provide a more personalized and engaging user experience.

As people discuss a variety of topics on our platform, many different opinions are voiced in the form of upvotes, downvotes, and replies (see Fig. 1). However, typing a reply on a phone is not the most convenient experience, so we have started experimenting with the ability to provide highly personalized reply suggestions, with the hope that it will improve user experience and increase engagement on a per-yak basis as well as on the app overall.

While there has been prior work on using deep learning to

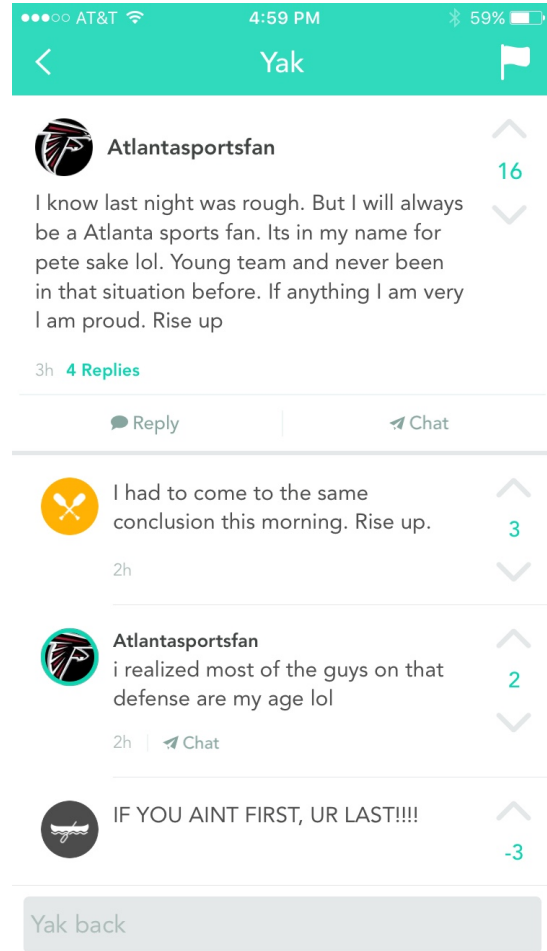


Figure 1: The Yik Yak app displaying a yak (post) along with replies and votes.

generate automated replies, these suggested responses tend to be overly generic, which gave us the impulse to build a novel and robust model that takes into account the location as well as user preferences in the generation of replies.

Let's say, as shown in Fig. 3, the question *What movie is good to watch on Netflix?* is posed. Does one answer makes sense for all users? Clearly not. Thus, we need to build intelligent agents that can be personalized per user or community to give more relevant and accurate responses.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...\$15.00.

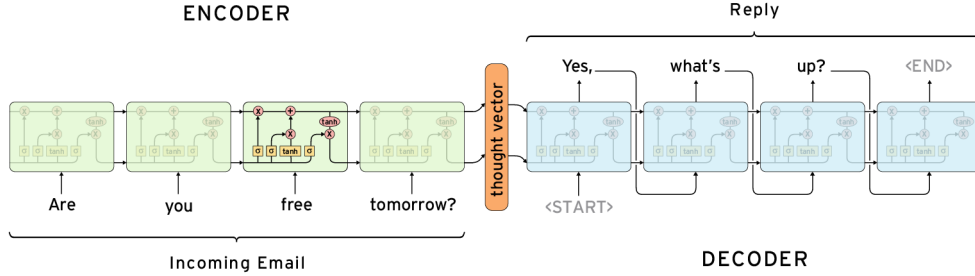


Figure 2: Google’s Smart Reply [11] using SEQ2SEQ.

What movie is good to watch on Netflix?

- **I like The Accountant** - Parry 😊 Robert 😊

Where do you want to party ?

- **Let's go to Vegas** Parry 😊 Robert 😊 Jacob 😊
- **I am down for Miami** Parry 😊 Robert 😊 Jacob 😊
- **We can go to LA** Parry 😊 Robert 😊 Jacob 😊

Figure 3: Examples of why a single suggested reply is not valid for an entire community and, instead, requires personalization.

The notion of persona or personality can change even for a single user over time or within the same day. For example, a college student may discuss class-related topics on weekday mornings but switch to talk about what events to attend over the weekend.

The coupling of text with demographic information has enabled computational modeling of linguistic variation, including uncovering words and topics that are characteristic of geographical regions [6]. In this paper, we take a step further and introduce a method that extends vector-space lexical semantic models to learn representations of geographically situated language. In bringing in extra-linguistic information to learn word representations, our work falls into the general domain of multimodal learning.

We present two models for personalization: first, a conversation model consisting of location- and user-based information, and then a social graph conversation model that combines the conversation models with social graph information. This represents, to our knowledge, the first time that a neural conversation model has been combined with a social graph to build a more intelligent and personalized agent, which can then be deployed in real-world applications such as highly-customized auto-reply suggestions.

## 2. RELATED WORK

There is a rich literature on the identification of important or influential nodes in a network, both in an unsupervised as well as semi-supervised manner. There has also been various work on conversation models or chatbots using neural generative models such as SEQ2SEQ. We briefly describe some of the related work in this section.

### 2.1 Conversation Models

Chatbots, also called conversational agents or dialog systems, have been studied by a variety of researchers from both academia and industry. We briefly describe the historical work and how they have evolved in time. There are two main classes of conversational models: retrieval-based and generative models.

#### RETRIEVAL-BASED MODELS

Retrieval-based models use a repository of predefined responses and some heuristic to choose an appropriate response based on the input and the context. The heuristic could be as simple as a rule-based expression match [7], or as complex as an ensemble of machine learning classifiers as in Lowe et al., 2015 [14]. These systems do not generate any new text and only pick a response from a fixed set.

#### GENERATIVE MODELS

Generative models go beyond predefined responses and are able to generate new responses from scratch. Such models are typically based on machine translation techniques, but rather than translating from one language to another, the input is translated into a response output. Here, we discuss SEQ2SEQ, one of the most promising generative models, which will become the baseline for our work.

### 2.2 SEQ2SEQ Models

A basic sequence-to-sequence model, as introduced in Cho et al., 2014 [4], consists of two recurrent neural networks (RNNs): an encoder that processes the input and a decoder that generates the output. Given a sequence of inputs  $X = \{x_1, x_2, \dots, x_n\}$ , an LSTM associates each time step with an input gate, a memory gate and an output gate denoted as  $i_t$ ,  $f_t$  and  $o_t$ , respectively. Furthermore,  $m_t$  represents the cell state vector at time  $t$ , and  $\sigma$  denotes the sigmoid function. Hence, the vector representation  $h_t$  for each time step  $t$  is given by:

$$\begin{aligned}
 i_t &= \sigma(W^u * h_{t-1} + I^u * x_t) \\
 f_t &= \sigma(W^f * h_{t-1} + I^f * x_t) \\
 o_t &= \sigma(W^o * h_{t-1} + I^o * x_t) \\
 c_t &= \tanh(W^c * h_{t-1} + I^c * x_t) \\
 m_t &= f_t \odot m_{t-1} + i_t \odot c_t \\
 h_t &= o_t \odot \tanh(m_t)
 \end{aligned} \tag{1}$$

In a SEQ2SEQ generation task, each input  $X$  is paired with a sequence of outputs to predict:  $Y = \{y_1, y_2, \dots, y_n\}$ . The LSTM defines a distribution over outputs and sequen-

<p>@reply_yak: Wanna party ?</p> <p>reply_yak APP 3:10 PM</p> <p>I have fun to be there</p> <p>I don't have a frat</p> <p>I want to get a lot</p> <p>We can i go</p> <p>You can go to party</p> <p>I'm a girl btw</p> <p>I'm a girl too</p> <p>I have fun of course</p> <p>I'm a girl lol</p>	<p>arash 11:37 AM</p> <p>@reply_yak: anyone down to watch a movie?</p> <p>reply_yak APP 11:37 AM ☆</p> <p>I want to watch it !</p> <p>I want to watch it now</p> <p>I just want to watch it</p> <p>I watched it on episode</p> <p>I just want to know</p> <p>I got a good luck !</p> <p>I want to watch it lol</p> <p>I want to watch netflix</p> <p>I was on the best friends</p> <p>I thought it on the movie</p>	<p>parry 4:01 PM ☆</p> <p>@reply_yak: Wanna go out on a date ?</p> <p>reply_yak APP 4:01 PM</p> <p>Let's go to late night</p> <p>I don't worry about to meet me</p> <p>I have fun date op</p> <p>I have a date op</p> <p>Op is a date me</p> <p>I don't worry about to meet people</p> <p>Go out of course !</p> <p>I want a good luck though</p> <p>Let's go to late night !</p> <p>I want a good luck</p> <p>I need a girl !</p>
---	---	--

Figure 4: YikYak’s Smart Reply using SEQ2SEQ with attention, where *reply\_yak* is the Slack bot.

tially predicts tokens using a softmax function.

Smart reply [11] (Fig. 2) which is used in Google’s auto reply system for emails is one of the practical applications of SE2SEQ. Similar to other sequence-to-sequence models, the Smart reply system is built on a pair of recurrent neural networks, one used to encode the incoming email and one to predict possible responses. The encoding network ingests the words of the incoming email one at a time and produces a vector (a list of numbers). This vector, which Geoff Hinton calls a **thought vector**, captures the gist of what is being said while abstracting from the specific words used. While these models are fairly easy to train, they have a tendency to produce generic answers for any type of topic or interaction. Also, they suffer from the same issues as general recurrent models, namely a vanishing gradient when the length of the sentence is too long. An area of research has been **attention**-based models [1], which emulate how humans give more importance to certain words in a sentence.

Attention mechanism (Fig. 5) predicts the output using a weighted-average context vector and not just the last state. As an example, in the sentence: *What is good to watch on TV?* we give more emphasis to *watch* and *TV* as opposed to the other words.

For our baseline model, we trained an attention-based SEQ2SEQ model using Yik Yak post and reply data and deployed it as a bot on Slack for demo purposes. Fig. 4 shows examples of some of the replies from the model, with a beam search size of 200.

There has also been recent work that incorporates topic [18] as well as context [8] in the SEQ2SEQ models in order to generate topic-based responses. While these models do come up with responses that might belong to one domain, they raise the question of whether it is sufficient for an intelligent agent to generate a single answer per topic or context.

### 2.3 Personalized Response Generation

Fig. 6 depicts the process of using a persona for building

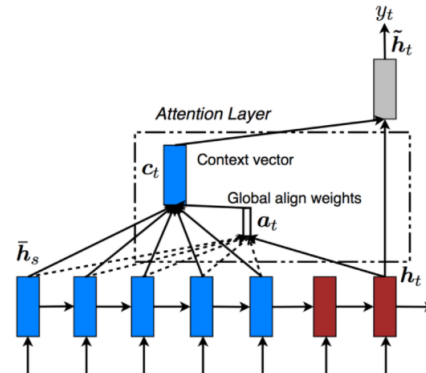


Figure 5: Attention-based model based on Bahdanau et al. 2015 [1].

a conversation model [12].

Li et al. [12] introduced a personalized conversation model that broadly consists of two models: the Speaker Model that integrates a speaker-level vector representation into the target part of the SEQ2SEQ model, and a Speaker-Addressee model that encodes the interaction patterns of two interlocutors by constructing an interaction representation from their individual embeddings and incorporates it into the SEQ2SEQ model. These persona vectors are trained on human-to-human conversation data and are used at test time to generate personalized responses.

### 2.4 Social Graphs and Embeddings

Various methods of graph embedding have been proposed in the machine learning literature (e.g., [2, 17, 5]). They generally perform well on smaller networks. The problem becomes much more challenging in a real-world information network with millions of nodes and billions of edges. In such networks, we seek to efficiently find low-dimensional embeddings that capture the network structure. Fig. 7 shows

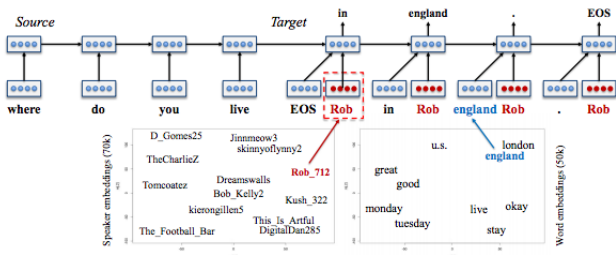


Figure 6: Example of a Speaker Model that integrates a speaker-level vector representation into the target part of the SEQ2SEQ model.

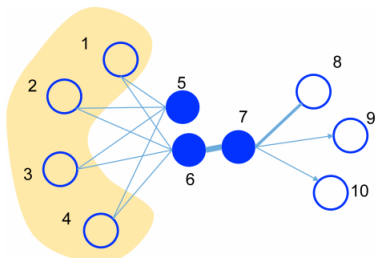


Figure 7: A toy example for an information network from LINE [16]. Edges can be undirected, directed, and/or weighted. Vertices 6 and 7 should be placed closely in the low-dimensional space as they are connected through a strong tie. Vertices 5 and 6 should also be placed closely as they share similar neighbors.

an illustrative example. Given that the weight of the edge between vertices 6 and 7 is large, i.e., they have a high first-order proximity, they should be represented closely to each other in the embedded space. At the same time, even though there is no direct link between vertices 5 and 6, they share many common neighbors, i.e., they have a high second-order proximity, and therefore should also be represented closely to each other. We expect that the consideration of the second order proximity effectively complements the sparsity of the first-order proximity and better preserves the global structure of the network.

One of the first efforts in this direction was **LINE**[16], a network embedding model, suitable for arbitrary types of information networks and efficiently scales to millions of nodes. The objective function is designed to preserve both the first-order and second-order proximities. The gradient will be multiplied by the weight of the edge. This will become problematic when the weights of edges have a high variance, but LINE paper avoids this issue by employing **edge sampling**, with the sampling probabilities being proportional to the original edge weights. Another approach is **Node2Vec** [9] which provides a flexible notion of neighborhood and employs a biased random walk to efficiently explore diverse neighborhoods.

### 3. TRAINING AND IMPLEMENTATION

Our work follows the footsteps of [12] work on persona-

based conversation model, which introduced the two persona-based models as previously discussed: the Speaker Model, which models the personality of the respondent, and the Speaker-Addressee Model which models the way the respondent adapts their speech to a given addressee.

#### 3.1 Training Protocols

The overall training procedure, used across all of the methods, is as follows:

- 4 layer LSTM model with 1,000 hidden cells for each layer.
- Batch size is set to 128.
- Learning rate is set to 1.0 with decay.
- Parameters are initialized by sampling from the uniform distribution  $[-0.1, 0.1]$ .
- Gradients are clipped to avoid gradient explosion with a threshold of 5.
- Vocabulary size is limited to 100,000.
- Dropout rate is set to 0.25.

#### 3.2 Decoding

For the decoding phase, the N-best lists are generated using the decoder with beam size  $B = 200$ . We set a maximum length of 20 for the generated candidates. Decoding is performed as follows: At each time step, we first examine all  $B \times B$  possible next-word candidates, and add all hypothesis ending with an EOS token to the N-best list. We then preserve the top- $B$  unfinished hypotheses and move to the next word.

#### 3.3 Dataset

The dataset used for the training consists of yak (post) and comment (reply) pairs. We preprocessed the pairs such that each post contains at least 5 words and no explicit language. After preprocessing, we obtained about 10 million pairs which were then randomly split into training and testing sets. This dataset encompassed 10,000 locations (at the county or city level) spread across 13 countries with a total of 100,000 unique users.

#### 3.4 Implementation

##### 3.4.1 Training

The source and target LSTMs use different sets of parameters. We ran 20 epochs, and training took roughly a week to finish on a g2.8xlarge AWS instance with 32 high frequency Intel Xeon E5-2670 (Sandy Bridge) processors as well as 4 high-performance NVIDIA GPUs, each with 1,536 CUDA cores and 4 GB of video memory.

##### 3.4.2 Inference

For inference, we used **Kubernetes**, an open-source system for automating deployment, scaling, and management of containerized applications. We used Kubernetes with **TensorFlow Serving**, a high-performance, open-source serving system for machine learning models, to meet the computational intensity and scaling demands of our application. The server executes the TensorFlow graph to process every

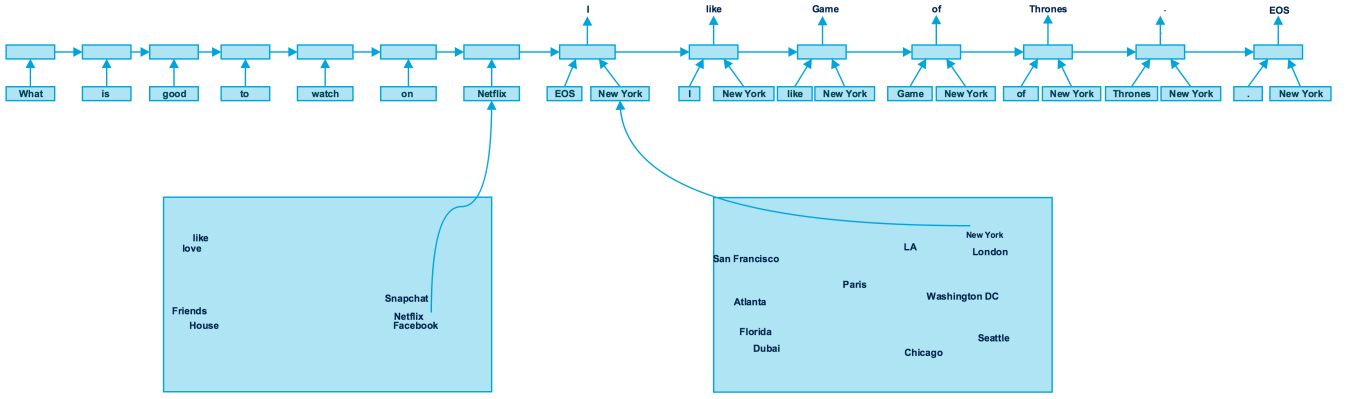


Figure 8: Location-based Conversation Model.

text suggestion request it receives. Kubernetes distributes the servicing of inference requests across a cluster using its External Load Balancer. Each pod in the cluster contains a TensorFlow Serving Docker image with the TensorFlow Serving Rest server and a trained SEQ2SEQ model. The model is represented as a set of files describing the shape of the TensorFlow graph, model weights, assets, and so on. Since everything is packaged together, we can dynamically scale the number of replicated pods using the Kubernetes Replication Controller to keep up with the service demands.

## 4. CONVERSATION-BASED MODEL

We introduce two types of conversation models: the **location based model**, which captures the community, and the **user-based model**, which is personalized for each user.

### 4.1 Location-based model

Given the location-based nature of Yik Yak, it is clearly important to incorporate location information in the model. Based on various independent studies, we have found out that the communities can vary significantly from each other in terms of both social connectivity and language usage, whereas the communities are internally homogeneous. In bringing in extra-linguistic information to learn word representations, our work falls into the general domain of multimodal learning. Unlike classic multimodal systems that incorporate multiple active modalities (such as gesture) from a user [15], our primary input is textual data, supplemented with the metadata about the author and the time of authorship. Thus, our first model is a location-based conversation model.

For this approach we developed two persona-based models: the **decoder model**, which captures the personality of the respondent, and the **encoder-decoder model**, which captures the way the respondents adapt their speech to a given addressee. Specifically, we use location embedding for both the encoder and decoder.

Given that each user event in our app is tagged with a latitude and longitude, we have a very robust way of understanding location. Here, we encapsulate the location information using three levels of granularity: county, city, and country. We concatenate the corresponding representations for each level to get the final local embedding (see Fig. 8).

The intuition behind this strategy is that if the data corresponding to a higher level of granularity is sparse, the lower level will provide a stronger signal.

As an example, the final local embedding for Queens county in New York is given as follows:

$$loc\vec{F}_{final}^{Queens} = [\vec{loc}^{Queens}, \vec{loc}^{NY}, \vec{loc}^{US}] \quad (2)$$

We used a final location embedding of size 300 in the model. As in the standard SEQ2SEQ models, we first encode the message  $S$  into a vector representation  $h_S$  using the source LSTM. Then, for each step in the target side, hidden units are obtained by combining the representation produced by the target LSTM at the previous time step, the word representations at the current time step, and the location embedding.

$$\begin{aligned} \mathbf{i}_t &= \sigma(\mathbf{W}^u * \mathbf{h}_{t-1} + \mathbf{I}^u * [x_t, loc\vec{F}_{final}^{Queens}]) \\ \mathbf{f}_t &= \sigma(\mathbf{W}^f * \mathbf{h}_{t-1} + \mathbf{I}^f * \mathbf{x}_t) \\ \mathbf{o}_t &= \sigma(\mathbf{W}^o * \mathbf{h}_{t-1} + \mathbf{I}^o * \mathbf{x}_t) \\ \mathbf{c}_t &= \tanh(\mathbf{W}^c * \mathbf{h}_{t-1} + \mathbf{I}^c * \mathbf{x}_t) \\ \mathbf{m}_t &= \mathbf{f}_t \odot \mathbf{i}_t \odot \mathbf{c}_t \\ \mathbf{h}_t &= \tanh(\mathbf{o}_t \odot \mathbf{m}_{t-1}) \end{aligned} \quad (3)$$

Here, we have provided the equations for a simple LSTM. In our final model, we used an attention-based model which outperforms the standard LSTM.

The LSTM defines a distribution over the outputs and sequentially predicts tokens using a *softmax* function. Since we want to predict the next word in a sentence, it would be a vector of probabilities across our vocabulary given by  $V$ .

$$\mathbf{o}_t = \text{softmax}(V * \mathbf{h}_t) \quad (4)$$

Finally, we minimize the average negative log probability of the target words:

$$loss = - \sum_{i=1}^N \ln p_{target} \quad (5)$$

### 4.2 User-Based Model

User-based models are similar to location-based ones with the difference that instead of using the location embeddings,



**Table 1: Perplexity of location- and user-based models.**

Model	Perplex.
LSTM (standard)	79.1
LSTM (attention)	77.2
Location-based model (decoder)	73.3
Location-based model (decoder and encoder)	<b>72.7</b>
User-based model (decoder)	79.6
User-based model (decoder and encoder)	<b>80.7</b>

we learn and use the user embeddings based on the conversational interaction between users. Compared with 10,000 location embeddings, we have about 100,000 user embeddings for the model to learn from.

### 4.3 Results and Discussion

The typical measure used for comparing different models is perplexity, defined as

$$e^{-\sum_{i=1}^N \ln p_{target}} = e^{loss} \quad (6)$$

In Table 1, we have summarized the results for our models and compared them with other techniques.

We observe that the location-based model results in a significant improvement (about 8% reduction in perplexity), whereas the user-based model is outperformed by the baseline (LSTM) models (about 2% increase in perplexity). The data sparsity for users may explain this observation, as there is less data per user compared with per location. Moreover, the number of embeddings to be learned for the user-based model (1,00,000) is 10 times more than the corresponding number for location-based model (10,000). We further observe that the decoder-and-encoder model performs worse than the decoder model, which shows that the speaker information does not lead to better suggestions (i.e., replies) in an anonymous environment.

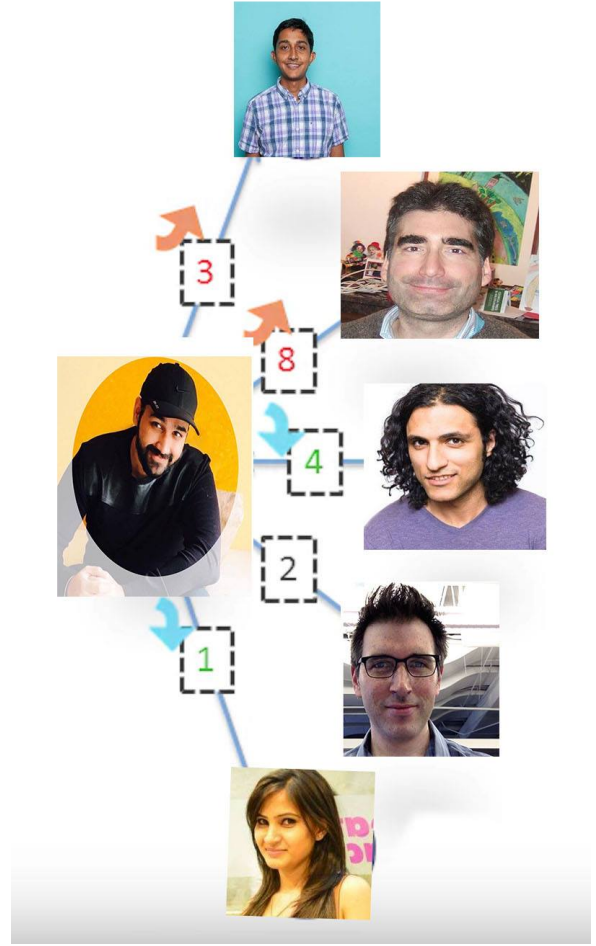
This motivated us to build a novel model that incorporates user information in a more robust way.

## 5. SOC2SEQ: SOCIAL-GRAPH USER MODEL

To overcome the deficiencies of the user-based model described in the previous section, we propose a joint model **soc2seq** whose objective combines **social embedding** with **conversation model**. Based on the results in user-based model, we have used only the decoder model. The proposed model works online without retraining the entire model and is robust to data sparsity. First, we discuss the social graph and how it is constructed at Yik Yak, and then we describe how the social graph can be used to learn low-dimensional embeddings and also explain how it can be combined with the prior models.

### 5.1 Interaction-based Social Graph

Social graphs have numerous applications and are considered the most important ranking factor for functionality such as people discovery (e.g., “people you may know”) and algorithmic content feed creation (e.g., “feed personalization”). As we do not have explicit social links such as friends (Facebook) or followers (Twitter) between users, we



**Figure 9: Weighted Social Graph based on Interactions.**

need to build the latent social graph based on the user interactions within our platform. The idea behind building this model comes from the fact that over time, people will develop a preference for certain users in their community.

Once we have general location-based social graph, we can apply deep learning models such as Node2Vec, DeepWalk, LINE, etc. to find such similarities among users. An important aspect to note is that in order to construct this model we did not make use of the textual content but rather we based it solely on user-to-user interactions.

### 5.2 Building interaction graphs

As mentioned above, one of the challenges we face at Yik Yak is the lack of an explicit notion of users being friends or following one another. Therefore, to build the social graph, we applied a bottom-up technique to leverage user interactions within the platform. Specifically, various interactions need to be combined and weighed to calculate the users affinity. The signals to be aggregated can be summarized in following categories:

- **Profile view** (directed and binary): This graph is based on whether a user has viewed another user’s profile.

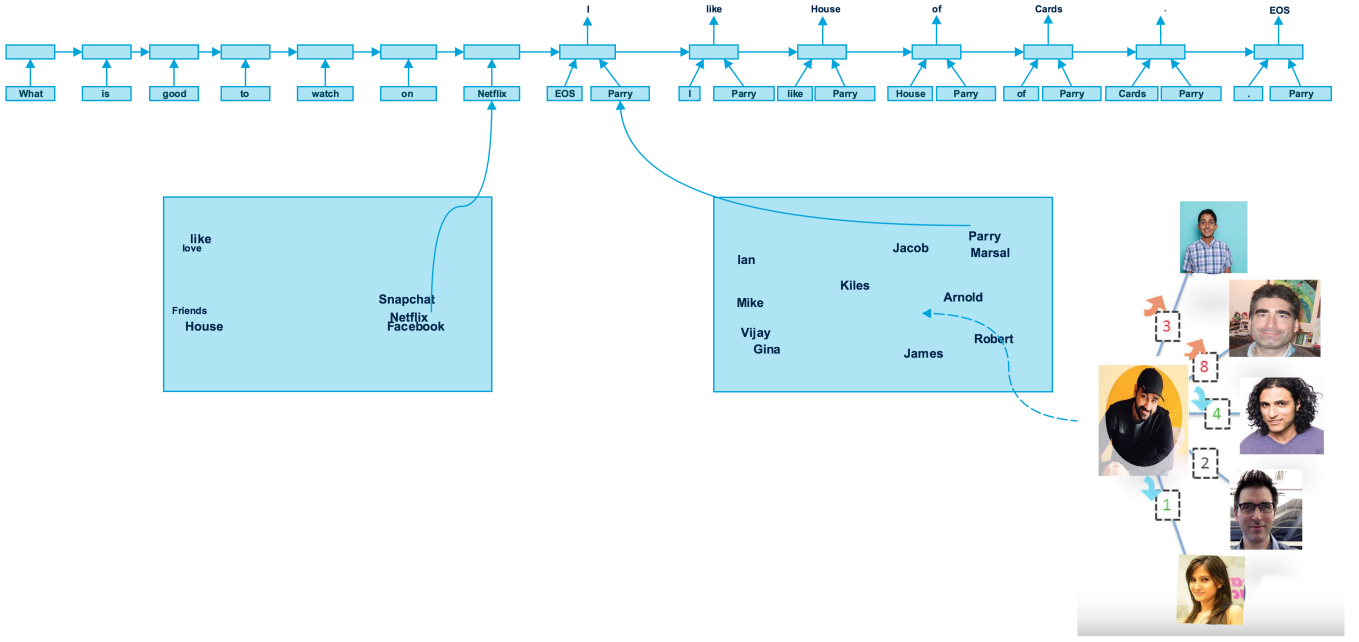


Figure 10: soc2seq: Social Conversation Model.

- **Chat request** (directed and binary): This graph is based on whether a user has sent chat request to other user.
- **Comment** (directed and weighted): The edges correspond to replies, where a user commented on another user’s post. The weight is determined by the number of such interactions.
- **Like** (directed): The edges correspond to upvotes (likes), where a user liked another user’s post. The weight is determined by the number of such interactions.
- **View** (directed-weak signal and weighted): The edges correspond to post views, where a user viewed another user’s yak or comment. The weight is determined by the number of such interactions.

In practice, different graphs are used for different applications. For example, for a user recommendation or “people you may know” feature, the objective is to maximize chat requests and profile views. On the other hand, for the construction of personalized feed, the objective function would be a weighted combination of all five graphs.

Here, we use Node2Vec [9] to obtain the social embedding for each user. Node2Vec optimizes the following objective function, which maximizes the log-probability of observing a network neighborhood  $N_S(u)$  for a node  $u$  conditioned on its feature representation as given by  $f$ ,

$$\max_f \sum_{u \in V} \log p(N_S(u) | f(u)) \quad (7)$$

which can be simplified to

$$\max_f \sum_{u \in V} \left[ -\log Z_u + \sum_{n_i \in N(u)} \log f(n_i) \cdot f(u) \right] \quad (8)$$

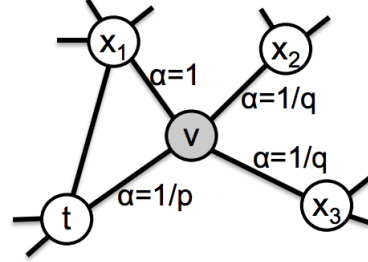


Figure 11: Illustration of the random walk procedure in Node2Vec. The walk just transitioned from node  $t$  to  $v$  and is now evaluating its next step out of node  $v$ . Edge labels indicate search biases  $\alpha$ . From [9].

Furthermore, negative sampling is used to speed up the process, instead of calculating a per-node partition function. The main challenge here is the definition of the neighbors. The neighborhoods  $N_S(u)$  are not restricted to the immediate neighbors and can have vastly different structures depending on the sampling strategy  $S$ .

In fact, the major advantage of the Node2Vec model is the flexible notion of the neighborhoods by designing a biased random walk to efficiently explore diverse neighborhoods. As shown in Fig. 11, the second-degree random walk in Node2Vec has two parameters,  $\mathbf{p}$  and  $\mathbf{q}$ . The **return parameter**  $\mathbf{p}$  controls the likelihood of immediately revisiting a node in the walk. Setting it to a high value ( $> \max(\mathbf{q}, 1)$ ) ensures that we are less likely to sample an already visited node in the following two steps (unless the next node in the walk has no other neighbor). This strategy encourages mod-

Table 2: Results using our soc2seq model.

Model	Perplex.
LSTM (standard)	79.1
LSTM (attention)	77.2
Location-based model (decoder)	73.3
Location-based model (decoder and encoder)	72.7
Social user model (standard)	72.4
Social user model (tuned)	<b>70.9</b>

erate exploration and avoids 2-hop redundancy in sampling. On the other hand, if  $\mathbf{p}$  is low ( $< \min(\mathbf{q}, 1)$ ), it leads the walk to backtrack a step (see Fig. 11) thus keeping the walk more local, i.e. close to the starting node  $u$ . On the other hand the **in-out parameter**,  $\mathbf{q}$  allows the search to differentiate between *inward* and *outward* nodes. Moreover, as seen again in Fig. 11, if  $\mathbf{q} > 1$ , the random walk is biased towards nodes close to node  $t$ . Such walks obtain a local view of the underlying graph with respect to the starting node in the walk and approximate BFS behavior in the sense that the sampled nodes tend to stay within a small locality.

In contrast, if  $\mathbf{q} < 1$ , the walk is more inclined to visit nodes that are further away from the node  $t$ . Such behavior is more similar to DFS, which encourages outward exploration. However, an essential difference here is that we achieve DFS-like exploration within the random walk framework. As such, the sampled nodes in Node2Vec are not at strictly increasing distances from a given source node  $u$ , but in turn, we benefit from tractable preprocessing and superior sampling efficiency of random walks.

Hence the overall joint loss function of our model is given by:

$$\begin{aligned} loss_{total} &= loss_{conversation} + loss_{social} \\ &= - \sum_{i=1}^N \ln p_{target} + SGD(node2vec\_walk) \end{aligned} \quad (9)$$

where SGD is the stochastic gradient descent on the Node2Vec random walk from equation 10.

Now, due to the training complexity of running a random walk on the entire graph for every conversation, we first trained Node2Vec from the interaction graph and then used those embeddings in the user-based conversation models, as depicted in Fig. 10. Moreover,  $\mathbf{p}$  and  $\mathbf{q}$  are set to 1, which were found to be the optimal values for downstream tasks such as chat link prediction.

Specifically for the reply suggestion task, we used a combination of the **comment** and **like** graphs, so the embedding for user **Alice** is given by:

$$\vec{u}_{ser\ Alice} = [\vec{comment\ Alice}, \vec{like\ Alice}] \quad (10)$$

As we can see in the results from Table 2, using pretrained embeddings from the like and comment views of the social graph boosts the results, even without pretraining the user embeddings. The reason for such a boost is due to the fact that using the interaction graph we can find clusters of users in the embedding space that reply in a similar fashion. Furthermore, a significant gain is observed when we fine tune the user embeddings by using both social and conversation information.

## 6. PRACTICAL OUTPUTS

Having measured the performance of the system from a perplexity standpoint, it is also important to observe how it performs in practice.

### 6.1 Location-Based Examples

We select five locations randomly for various posts to obtain the replies. Fig. 12 gives an example for the question *Anyone wanna watch Netflix* and shows that different answers can be generated based on the location. From the replies it can be observed that *Daredevil* is popular among New Yorkers while *Games of Thrones* is popular in London. Such flexibility could not have been possible in earlier models.

- **Anyone wanna watch netflix ?**
  - **New York** : Daredevil is lit !!!
  - **LA**: Lets go for House of Cards.
  - **SF**: orange is the new black 🍊 🍊
  - **London**: GOT
  - **Atlanta**: netflix and chill
- **I am feeling lonely and depressed**
  - **New York** : Lets party.
  - **LA**: Find a date dude.
  - **SF**: we are there op
  - **London**: I think of you as friend
  - **Atlanta**: Are you a girl ?

Figure 12: Sample output of the location-based conversation model.

### 6.2 Social-Graph-Based Examples

For users, we randomly selected five people from the 100,000 users and evaluated their response on different sets of posts. As observed in Fig. 13, each of the have a unique but consistent personality, as observed in the way they reply. For example, User1 appears to be extrovert, and User4 is likely a female.

## 7. CONCLUSION AND FUTURE WORK

We have presented a novel approach that jointly models the conversational and social aspects of user interactions. This model allows for intelligent agents that learn from both the content as well as the structure of user interactions and thus better emulate personalized behavior. It achieves a substantial gain in perplexity for location-based as well as social-based models. We have demonstrated that by encoding personas in distributed representations of conversation and social graphs, one can capture personal characteristics such as speaking style and background information.

This model represents a building block for future work, which includes making it more robust to unknown words by



- **I wanna start gym**

- **User1** : good luck op.
- **User2**: I am too fat 🍔
- **User3**: you are the man.
- **User4**: I want to start too.
- **User5**: Go out ofcourse.

- **I love you**

- **User1** : I love you too babe.
- **User2**: I have a boy friend 🍷
- **User3**: I am a boy.
- **User4**: I am a girl lol.
- **User5**: Are you a girl ?

**Figure 13: Sample output of the soc2seq social conversation model.**

incorporating morphemes [3] or even character-level embeddings. Also we plan to explore combining social graph with Li et al. 2016 approach [13] of using reinforcement learning as delayed rewards and policy gradient in order to incorporate diversity, ease of answering, and enforce semantic coherence. Yet another area might be to make sequential networks such as LSTM more intelligent as in HyperNetworks [10], where a smaller network helps the main network to make intelligent decisions.

## 8. ACKNOWLEDGMENTS

We thank Vijay Viswanathan, Yi Yang, Jacob Eisenstein, and Tomasz Jurczyk for their helpful discussion of this work.

## 9. REFERENCES

- [1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015.
- [2] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *NIPS*, 14:585–591, 2001.
- [3] P. Bhatia, R. Guthrie, and J. Eisenstein. Morphological priors for probabilistic neural word embeddings. *EMNLP*, 2016.
- [4] K. Cho, B. Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv:1406.1078*, 2014.
- [5] T. F. Cox and M. A. Cox. Multidimensional scaling. *CRC*, 2000.
- [6] J. Eisenstein, B. O’Connor, N. A. Smith, and E. P. Xing. A latent variable model for geographic lexical variation. *EMNLP*, 2010.
- [7] M. Galley, C. Brockett, A. Sordani, Y. Ji, and M. Auli. A discriminative metric for generation tasks with intrinsically diverse targets. *arXiv:1506.06863*, 2015.
- [8] S. Ghosh, O. Vinyals, B. Strope, S. Roy, T. Dean, and L. Heck. Contextual lstm (clstm) models for large scale nlp tasks. *arXiv:1602.06291*, September 2016.
- [9] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. *KDD*, August 2016.
- [10] D. Ha, A. Dai, and Q. Le. Hypernetworks. *arXiv:1609.09106*, September 2016.
- [11] A. Kannan, K. Kurach, S. Ravi, T. Kaufmann, A. Tomkins, and B. Miklos. Smart reply: Automated response suggestion for email. *KDD*, August 2016.
- [12] J. Li, M. Galley, C. Brockett, G. Spithourakis, J. Gao, and B. Dolan. A persona-based neural conversation model. *ACM Trans. Program. Lang. Syst.*, 1(5):994–1003, 2016.
- [13] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky. Deep reinforcement learning for dialogue generation. *EMNLP*, 2016.
- [14] R. Lowe, N. Pow, I. V. Serban, and J. Pineau. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *SIGDIAL*, 2015.
- [15] S. Oviatt. *The Human-computer Interaction Handbook*. Julie A. Jacko and Andrew Sears, Hillsdale, NJ, USA, 2003.
- [16] J. Tang, M. Qu, and M. Wang. Line: Large-scale information network embedding. *WWW*, 2015.
- [17] J. B. Tenenbaum, V. D. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500), pages 2319–2323, 2000.
- [18] C. Xing, W. Wu, Y. Wu, and J. Liu. Topic aware neural response generation. *arXiv:1606.08340*, September 2016.